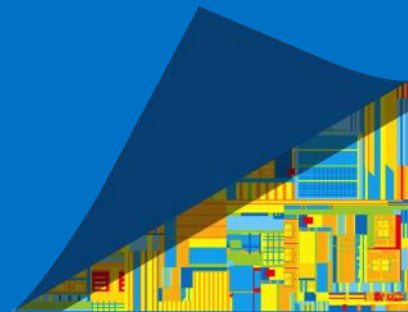




Using the Second-Screen API and Intel® Wireless Display From Android* apps.

Xavier Hallade, Technical Marketing Engineer



What is Miracast*?

Miracast* is standard for wireless peer-to-peer screen-casting, created and certified by the WiFi Alliance*

Wireless replacement for HDMI*/MHL*

Built upon existing standards

- H.264 (MPEG-4 AVC) video compression & WiFi Direct

Based on Intel® Wi-Fi Display specifications

Open for the industry

Support added in Android* 4.2

It is NOT DLNA or ChromeCast





What is Intel® Wireless Display (WiDi)

- WFA Miracast* certified
 - compatible with Miracast* certified receivers.
- Supports HD video up to 1080p/60 with 5.1 audio
- Content protection with HDCP2.1

Supports two additional Display modes

- Extended Video mode (with Android* media framework)
- Multi-task usage support (content on second screen, while local screen used for other task)

Available on *most* Intel® Atom™ SOC



*Intel® WiDi is a superset of WFA Miracast**

Miracast* Concept

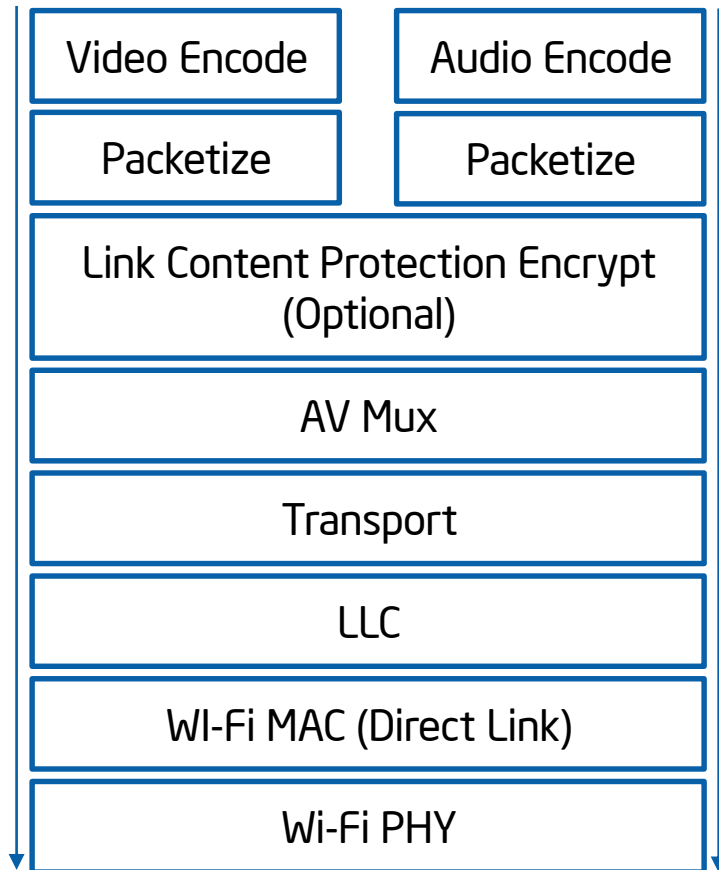


SOURCE

Video Frames



Audio Samples

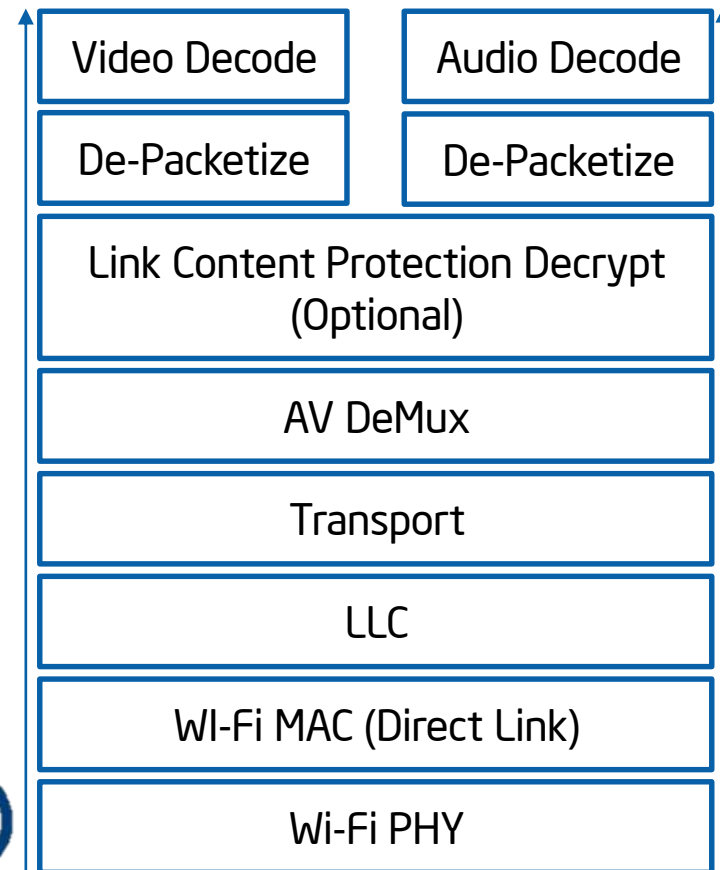


SINK

Video Render



Audio Render



Miracast* Certified Sink Devices

Adapters

- Actiontec* ScreenBeam Pro
- Netgear* (ex: PTV3000)
- Viewsonic (WPG-370)
- Acer*
- Best Buy* Rocketfish*
- Lenovo*
- ...



Actiontec®



NETGEAR®
Connect with Innovation™

Intel® WiDi Certified

Projectors

- Dell*
- Seiko Epson *
- Ricoh*
- LG*
- ...



TVs

- Samsung*
- Toshiba*
- LG*
- TCL*
- Sharp*
- Philips*
- ...



TOSHIBA
Leading Innovation >>>



Full list of Miracast* certified devices can be found at
<http://www.wi-fi.org/wi-fi-certified-miracast>

Miracast vs Intel® WiDi Receiver Spec



Features	Miracast	Intel® WiDi (4 th Gen)
Resolution	480p/30fps 720p Optional	1080p/30fps 1080p/60fps Optional
HDCP	Optional	2.1
Audio	Stereo Sound	Stereo Sound & 5.1 Surround Sound
S3D	Optional	Yes
Speed Mode	Optional	Low Latency
Fast Cursor	No	Yes

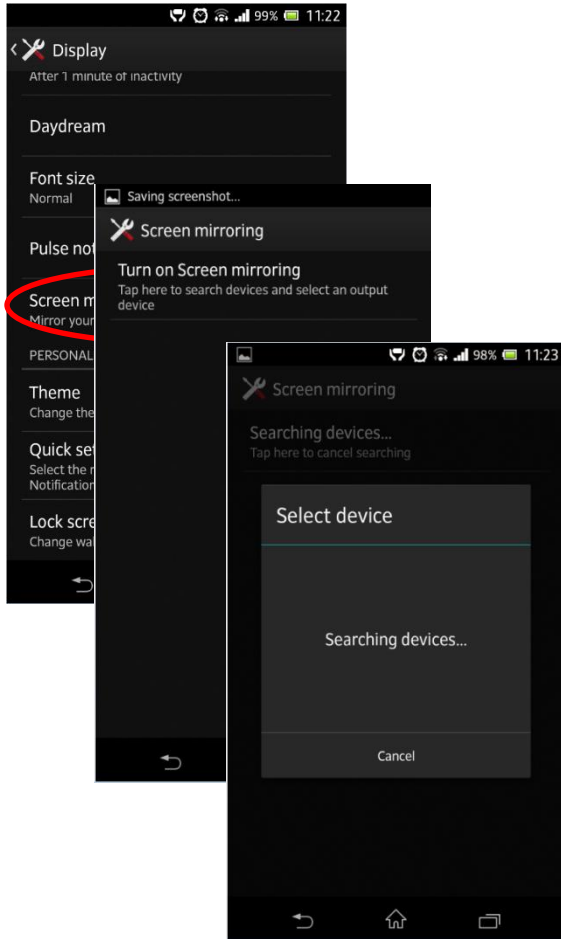
Second-Screen Enabled Devices running Android^{*}

4.2.x

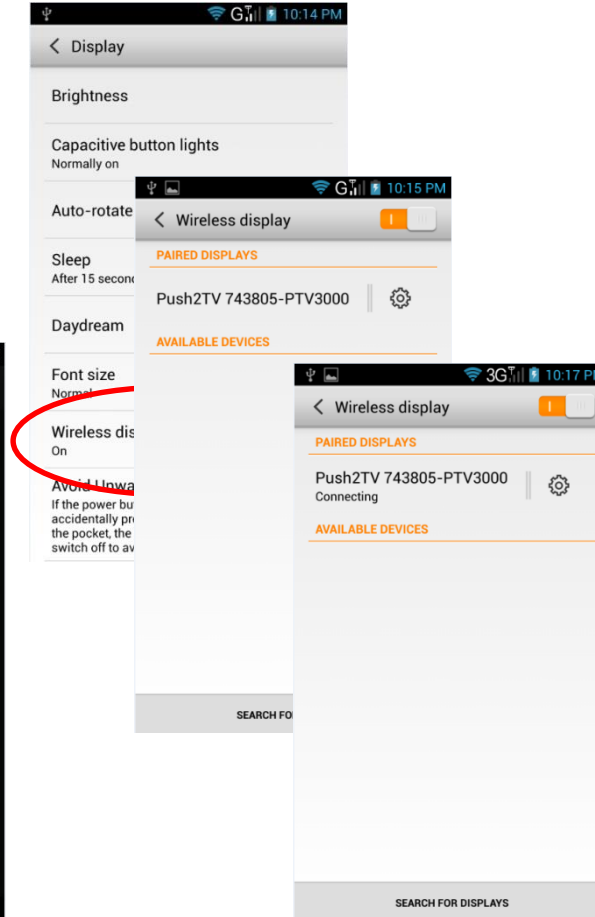
Device	Miracast [*] Certified	HDMI [*] , MHL [*] or SlimPort [*]
Samsung [*] Galaxy S4	Yes	Yes
Samsung Galaxy Tab3 10.1	No	Yes
Sony [*] Xperia Z	Yes	Yes
LG [*] Optimus G	Yes	Yes
Nexus [*] 4, 5	Yes	Yes
HTC [*] One	Yes	Yes
Lenovo [*] K900	Yes	No
Asus [*] MeMO Pad FHD10	Yes	Yes



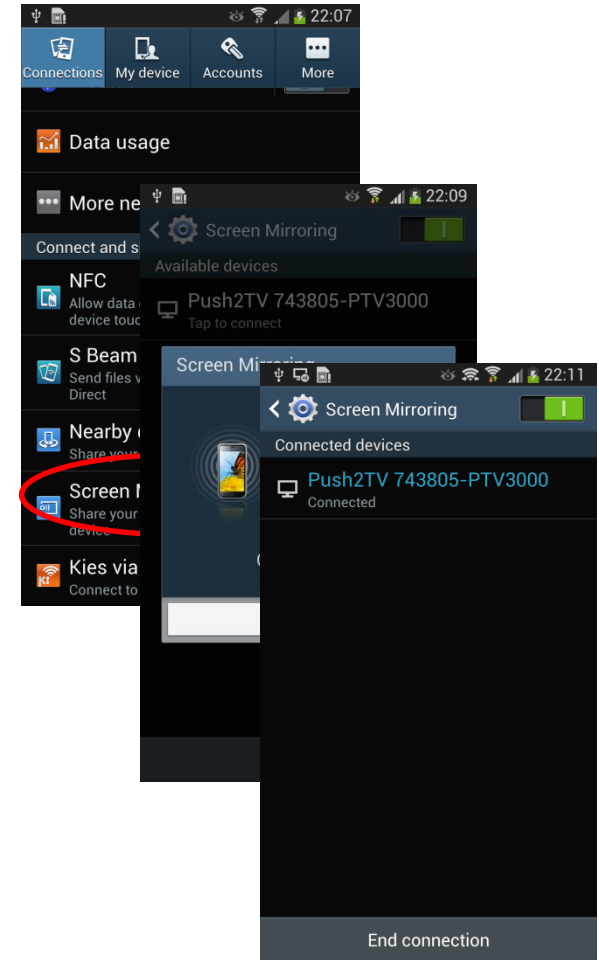
Connecting a Wireless Display on Android*



Sony* Xperia Z



Stock/Intel®



Samsung* Galaxy S4

Clone Mode (Default)



After establishing the connection, user sees local screen on the remote display

Resolution sent to remote is the same as local display's

No need to do anything to support this mode

Dual Screen Display With Single App



Remote screen used for content viewing

Local screen used for control & context info

Application can target this mode using the Android*
Presentation API

Extended Video Mode (Intel® Platform Specific)



Video mode is activated automatically when user plays a video using Android* Media Player framework (ex: VideoView)

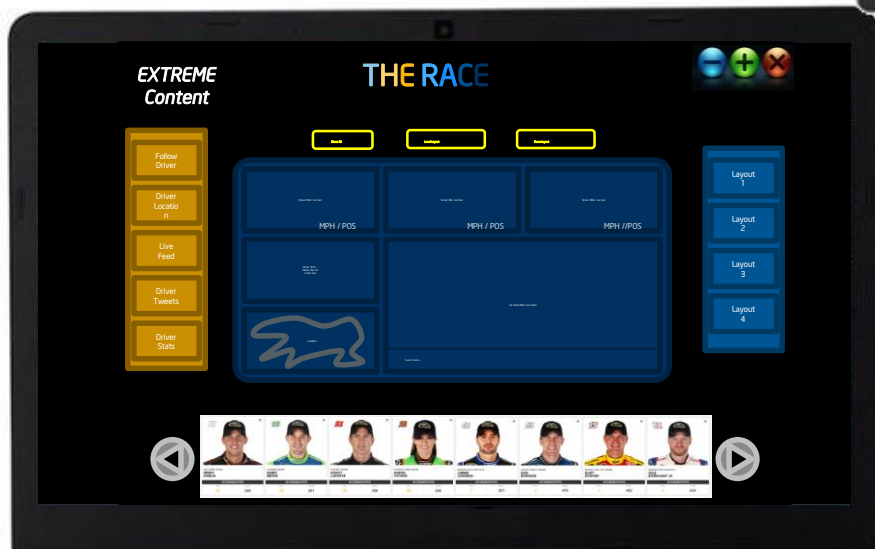
User sees video content on the remote at the 1080p resolution (or whatever the native resolution of the content is)

Local video rendering is turned off to save power, but UI stays untouched

Intel® WiDi – *Dual Screen Possibilities*

Multi-video User configured Application

App Window #1
Configure and add select
content
on 2' screen

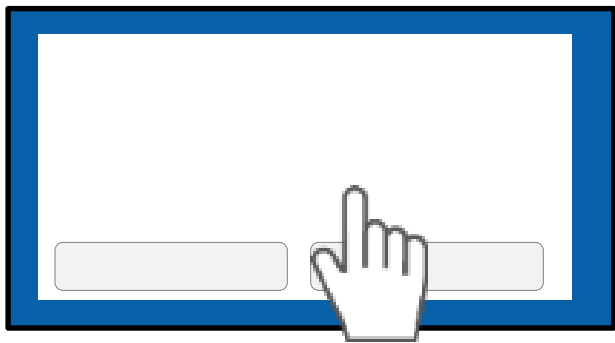


App Window #2
View multi-angle
Videos & more on
10' large screen

One application, pulling content from one site, driving two screens!

Ideas for Dual Screen Applications

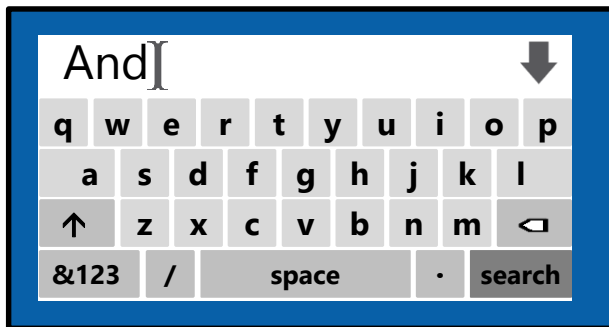
Web Browser



Touchpad Mode

Ideas for Dual Screen Applications

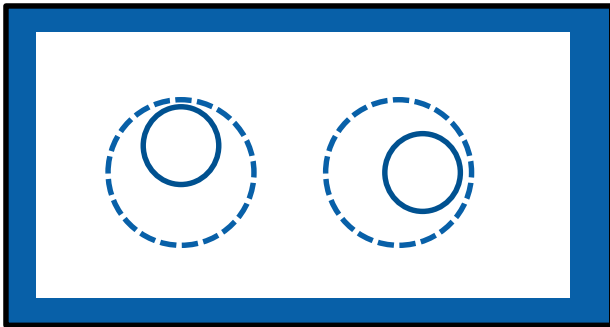
Web Browser



Keyboard Input Mode

Ideas for Dual Screen Applications

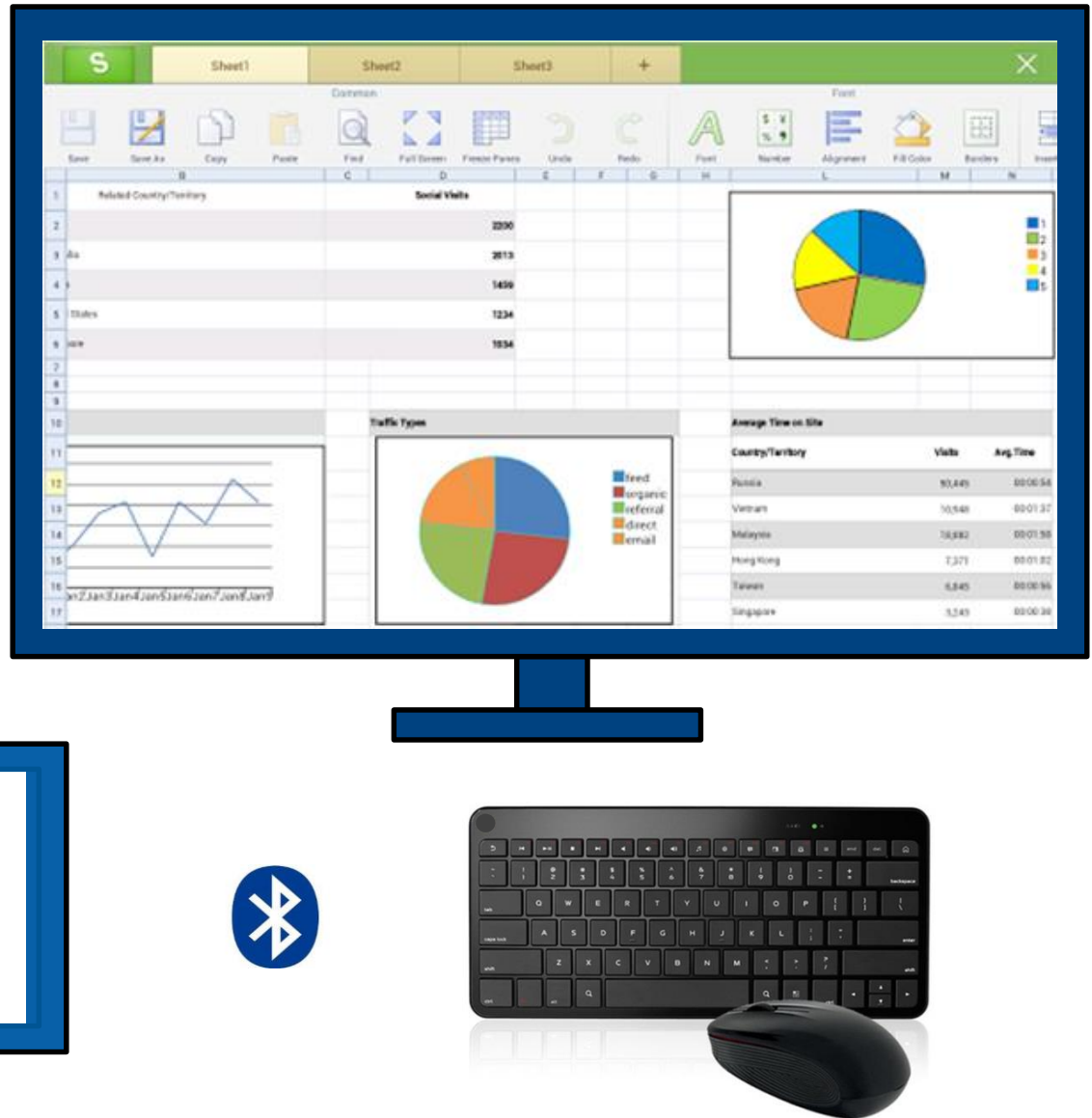
Games



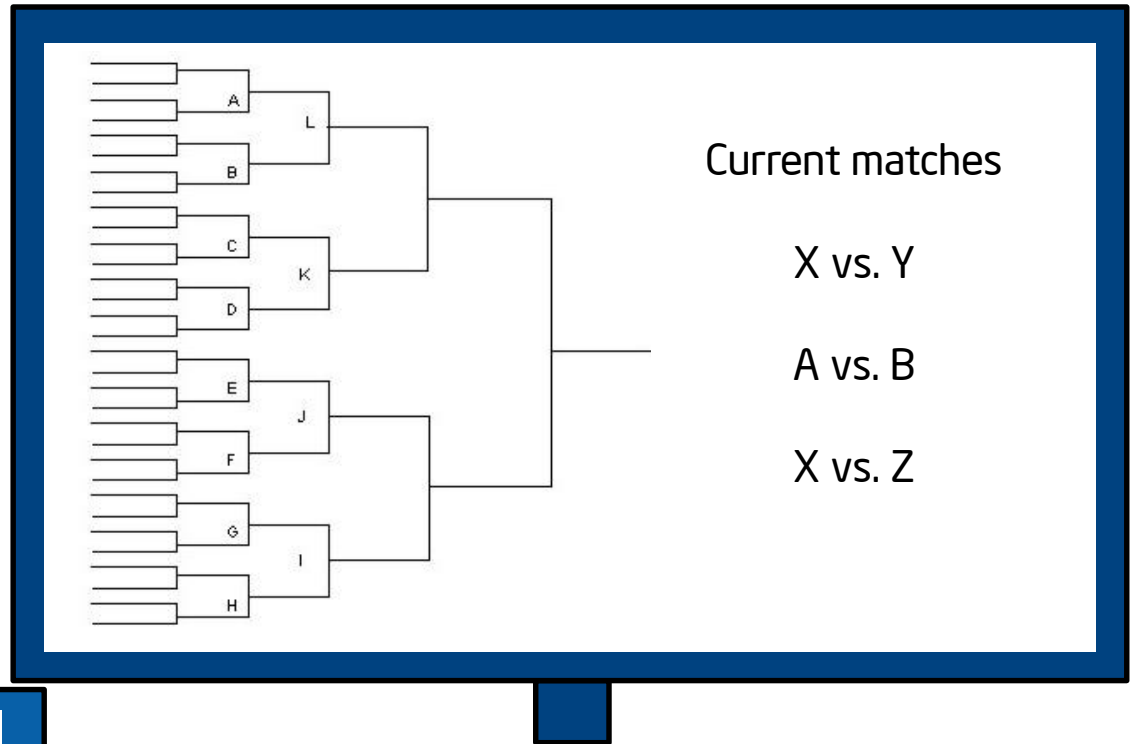
Dual Joysticks Mode

Ideas for Dual Screen Applications

Productivity App



Ideas for Dual Screen Applications



The small monitor displays the score for Player X vs. Player Z. Player X has a score of 11 and Player Z has a score of 7. There is an 'Add Score' button below the scores.

Player X vs. Player Z

11 7

Add Score

Tournament Manager

Second-Screen Demo

Android* Secondary Display API

Android added Second-Screen support via the ***Presentation*** class in Android* 4.2 (API Level 17), allowing you to:

- Implement support for a second screen in your applications without having to worry about the way the displays are connected (Display agnostic)
- Works with MHL*, HDMI*, SlimPort* & Miracast* compatible devices
- You can control the output on the remote (second) screen independently of the phone screen

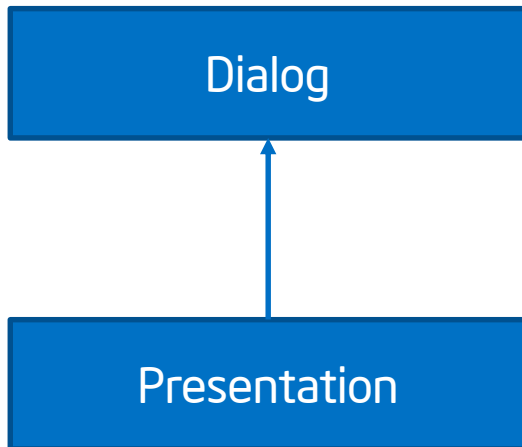


The Presentation Object

- Presentation is the based class and should be extended:

```
public class DemoPresentation extends Presentation {
```

- Needs to be associated with a Display at creation time
- Presentation inherits from Dialog, and as for a Dialog its lifecycle is bound to an Activity



You need to have a fragment based navigation if you want to keep it running while navigating in the app.

The activity should take care of pausing and resuming whatever content is playing within the presentation whenever the activity itself is paused or resumed.

Using the Presentation API

Before showing a Presentation you need to select a Display, this can be done in 2 ways:

1. **MediaRouter API** (in API 16): system will decide the best display for you!

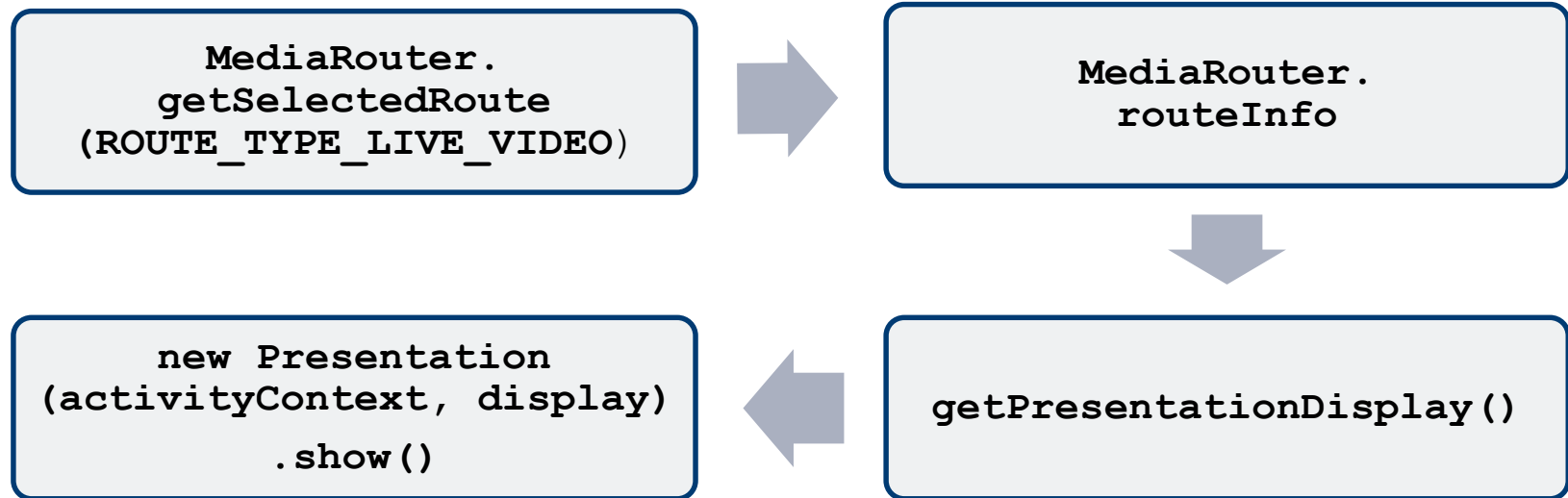
```
// Get the MediaRouter service
MediaRouter mMediaRouter =
    (MediaRouter)getSystemService(Context.MEDIA_ROUTER_SERVICE);
// Care only about routes that have full video support.
MediaRouter.RouteInfo mRouteInfo =
    mMediaRouter.getSelectedRoute(MediaRouter.ROUTE_TYPE_LIVE_VIDEO);
Display presentationDisplay = mRouteInfo.getPresentationDisplay();
```

2. **Display Manager API** (in API 17): Enumeration of displays

```
// Get the DisplayManager service.
DisplayManager mDisplayManager =
    (DisplayManager)getSystemService(Context.DISPLAY_SERVICE);
// enumerate the displays
Display[] presentationDisplays =
    mDisplayManager.getDisplays(DisplayManager.DISPLAY_CATEGORY_PRESENTATION);
```

Presentation API

How to get a Presentation displayed:



Then, using `MediaRouter.addCallback`, you have to monitor:

- `onRouteUnselected`
- `onRouteSelected`
- `onRoutePresentationDisplayChanged`

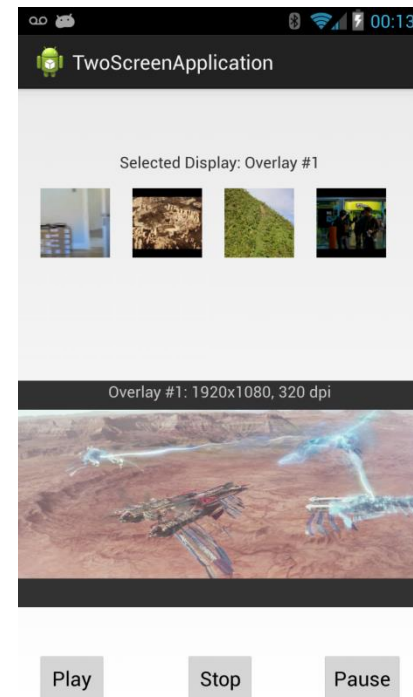
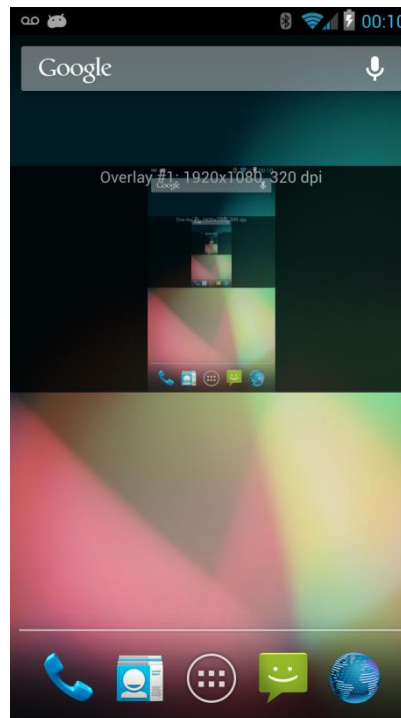
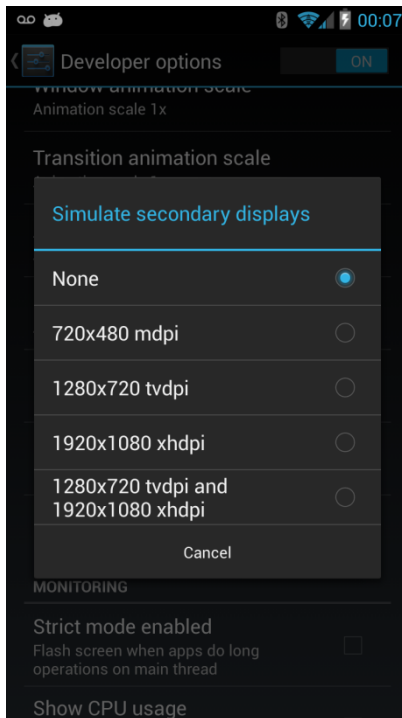
And inside the activity owning the Presentation:

- `onResume`
- `onPause`

Testing your Second-Screen Apps

Options:

- Real Hardware - supporting second screen functionality
- Simulate secondary screen - new feature in Android* 4.2 (in Developer options). You can select by resolution and DPI and once activated you get an overlay emulating the secondary screen.

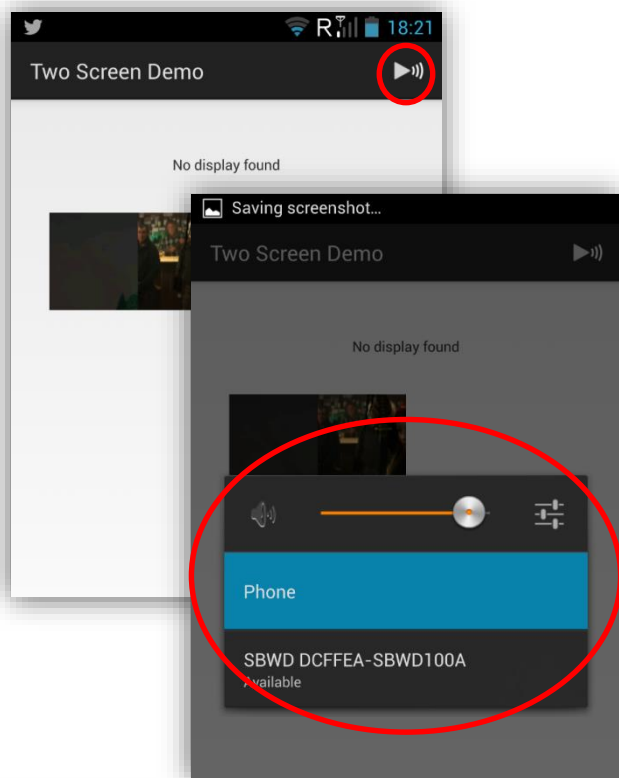


Adding a MediaRouteButton

In res/menu/default.xml:

```
<item  
    android:title="Media Route Settings"  
    android:actionProviderClass="android.app.MediaRouteActionProvider"  
    android:showAsAction="always" />
```

```
//Sets Media Route Button to second screen mode  
mediaRouteActionProvider.setRouteTypes  
    (MediaRouter.ROUTE_TYPE_LIVE_VIDEO);
```



Q&A